Firmado de zonas con el HSM distribuido

Eduardo Riveros
eduardo@niclabs.cl

https://niclabs.cl/tchsm

Temario

1. **Contexto**

   DNSSEC
   Proceso de firmado con OpenDNSSEC
   HSMs

2. **HSM Distribuido**

   Criptografía de Umbral
   HSM Distribuido
   Historia
   Componentes
   Características

3. **Instalación y Uso**

   Demostración en vivo

NIC CHILE RESEARCH LABS

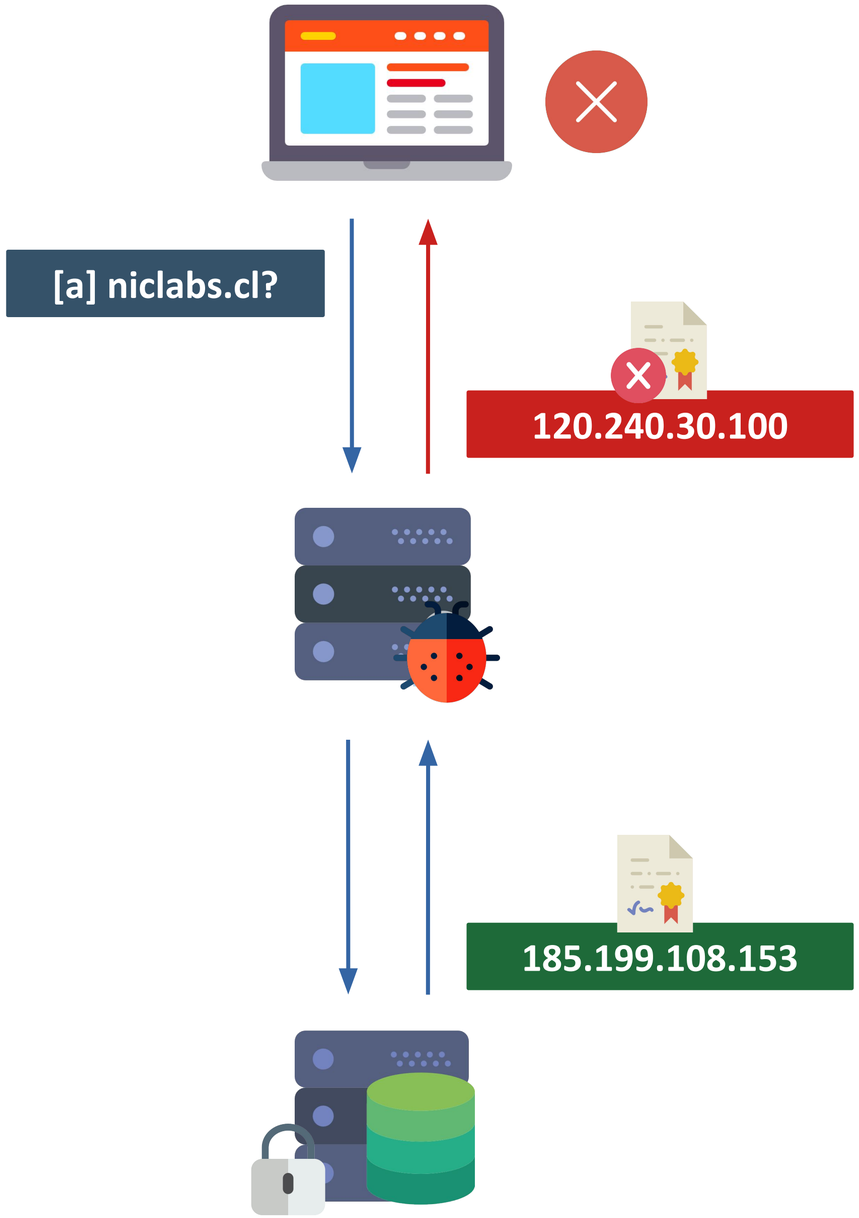
Contexto

NIC CHILE RESEARCH LABS

DNSSEC

Extensiones de Seguridad de DNS
Proveen a DNS de las siguientes características de seguridad:

- Autenticación de origen para datos recibidos
- Integridad de datos
- Prueba de no existencia autenticada



NIC CHILE RESEARCH LABS

Consulta DNSSEC

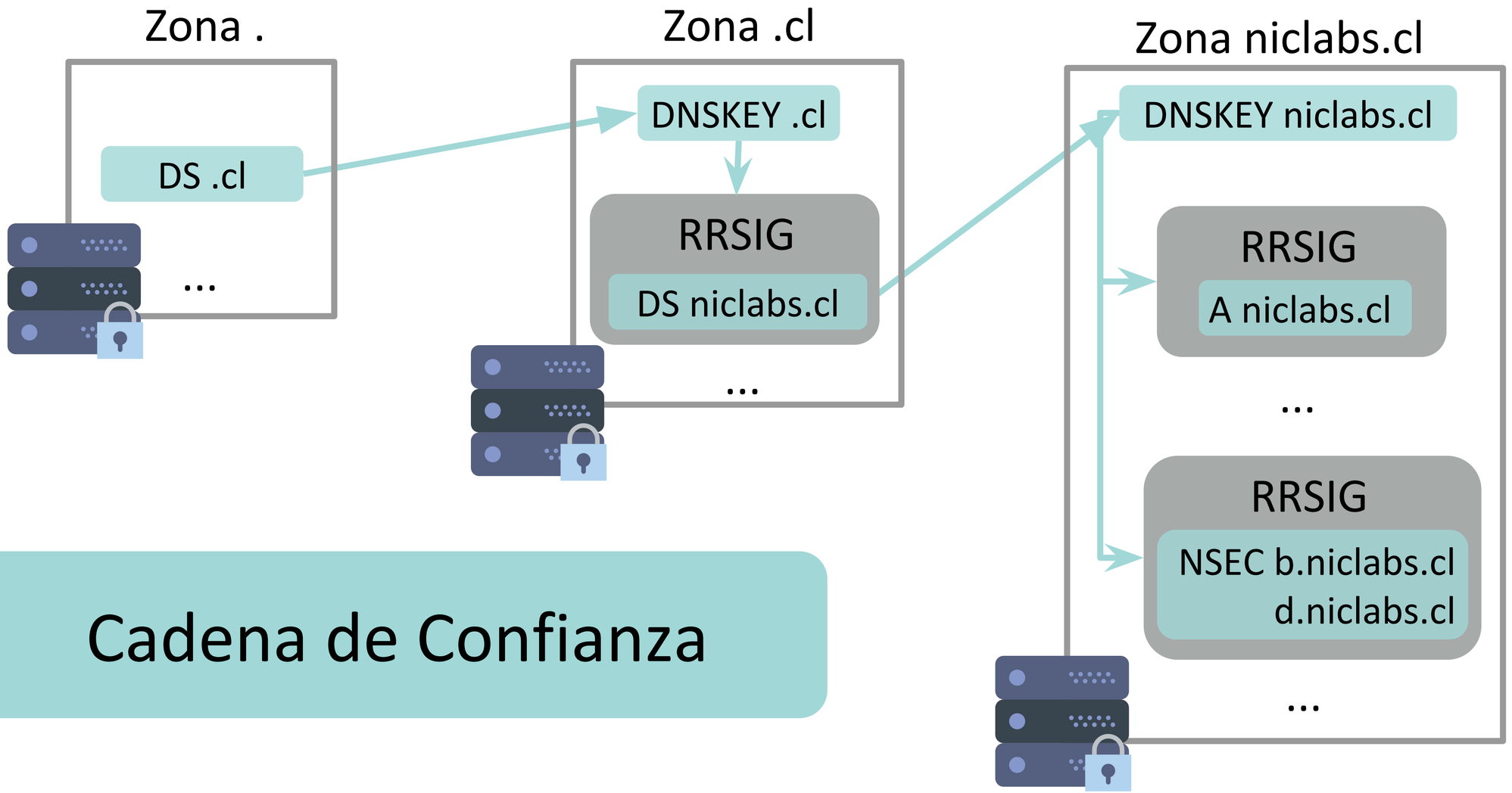
- Consulta usando flag DO (DNSSEC OK)
- Respuesta con flag AD (Authenticated Answer)

DS	Hash que valida la llave de DNSKEY. Presente en zona padre.
DNSKEY	Llave pública usada para validar firmas RRSIG.
RRSIG	Firma sobre un RRSet. Se valida con la DNSKEY.
NSEC(3)	Demostración firmada de inexistencia de cierto subdominio.

NIC CHILE RESEARCH LABS

¿Cómo funciona?

(muy simplificado)

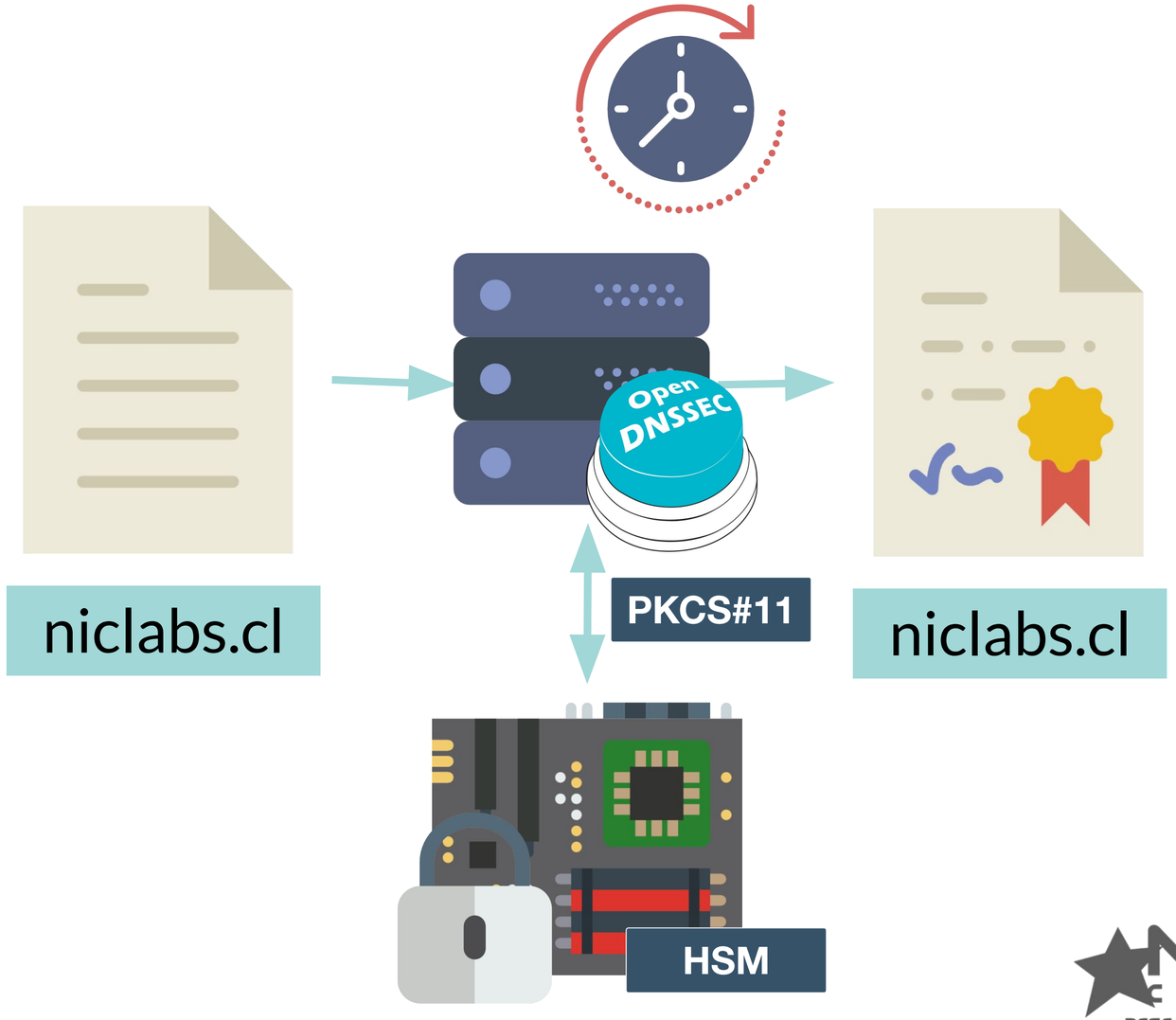


Cadena de Confianza

NIC CHILE RESEARCH LABS

OpenDNSSEC

- Sistema de código abierto de Firmado de zonas DNS.
- Maneja automáticamente el firmado de las zonas cuando es requerido.



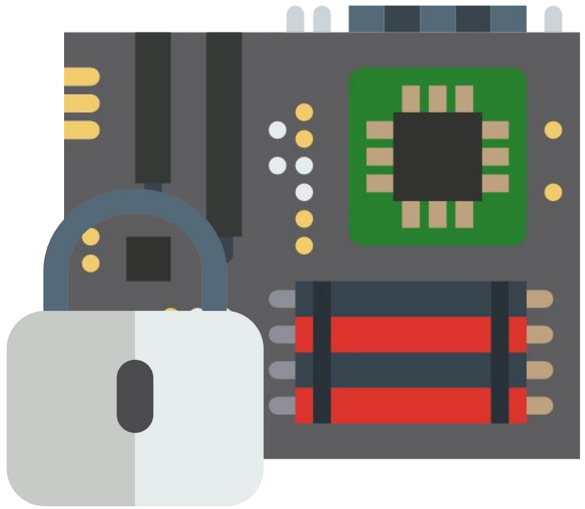
NIC CHILE RESEARCH LABS

~~HSM~~ para firmado de zonas

Hardware externo al equipo, especializado en **operaciones criptográficas**

Usable a través de una librería que cumple la interfaz **PKCS#11**

Existen implementaciones de Software (**SoftHSM**)



Alto Precio

Único punto de fallo

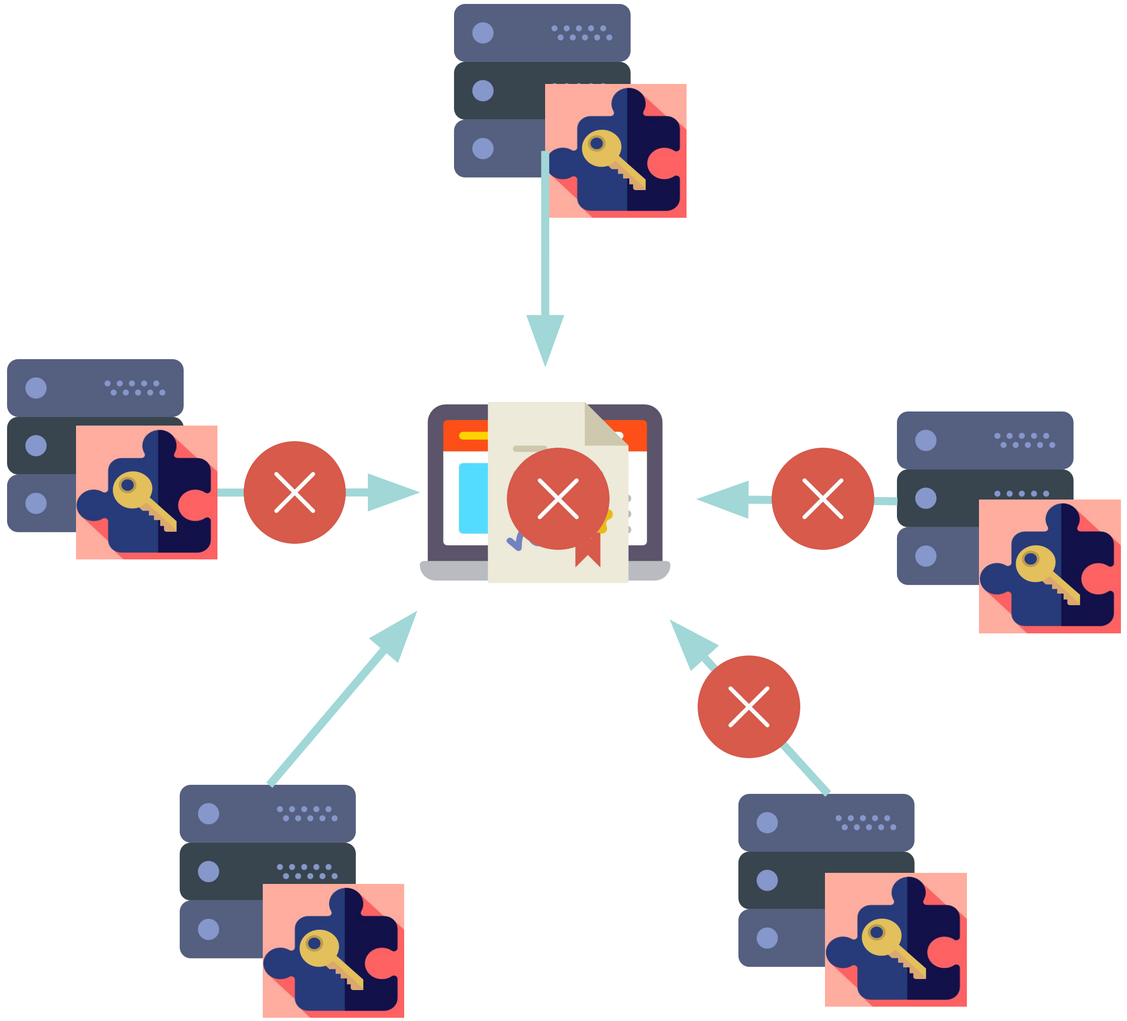
NIC CHILE RESEARCH LABS


HSM Distribuido

Criptografía de Umbral

Permite utilizar una llave criptográfica contando con la mayoría de trozos previamente repartidos.

Si una pequeña cantidad de trozos no son recuperables, el sistema sigue siendo funcional.



NIC CHILE RESEARCH LABS

dHSM

Distributed Hardware Security Module

Conjunto de Librerías que se comportan como un HSM, aprovechándose de la Criptografía de Umbral mediante el uso de varios nodos independientes.

Basado en paper de V. Shoup *“Practical Threshold Signatures”*

Cumple con API PKCS#11 *(OpenDNSSEC, KNOT)*

ØMQ

Comunicación entre nodos y librería usando ZeroMQ

C++ GO

Implementaciones en C++ y Go

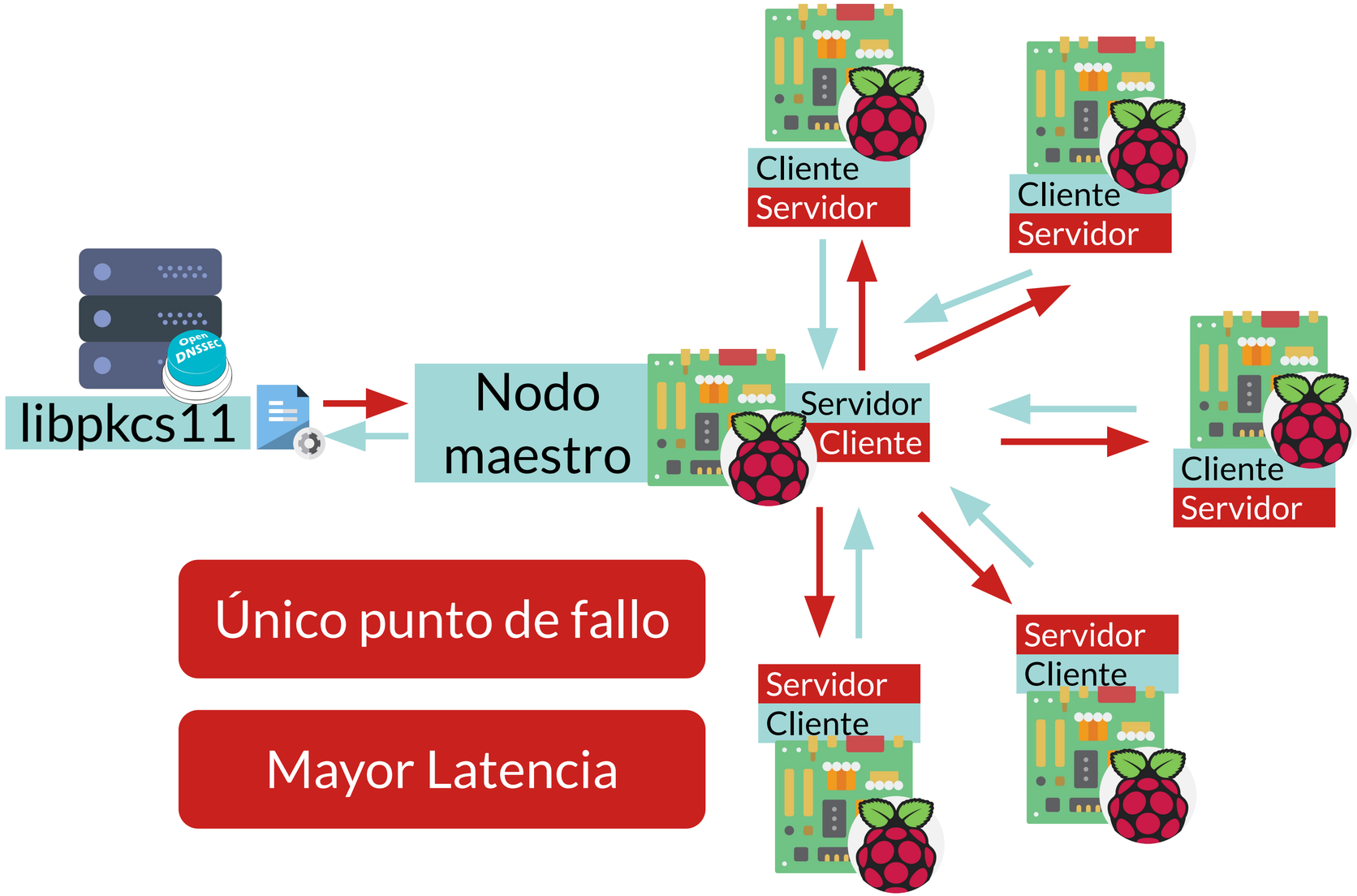
Implementación relativamente barata.

NIC CHILE RESEARCH LABS

Historia

2014

Implementación Inicial en C++ con un nodo maestro



Único punto de fallo

Mayor Latencia

NIC CHILE RESEARCH LABS

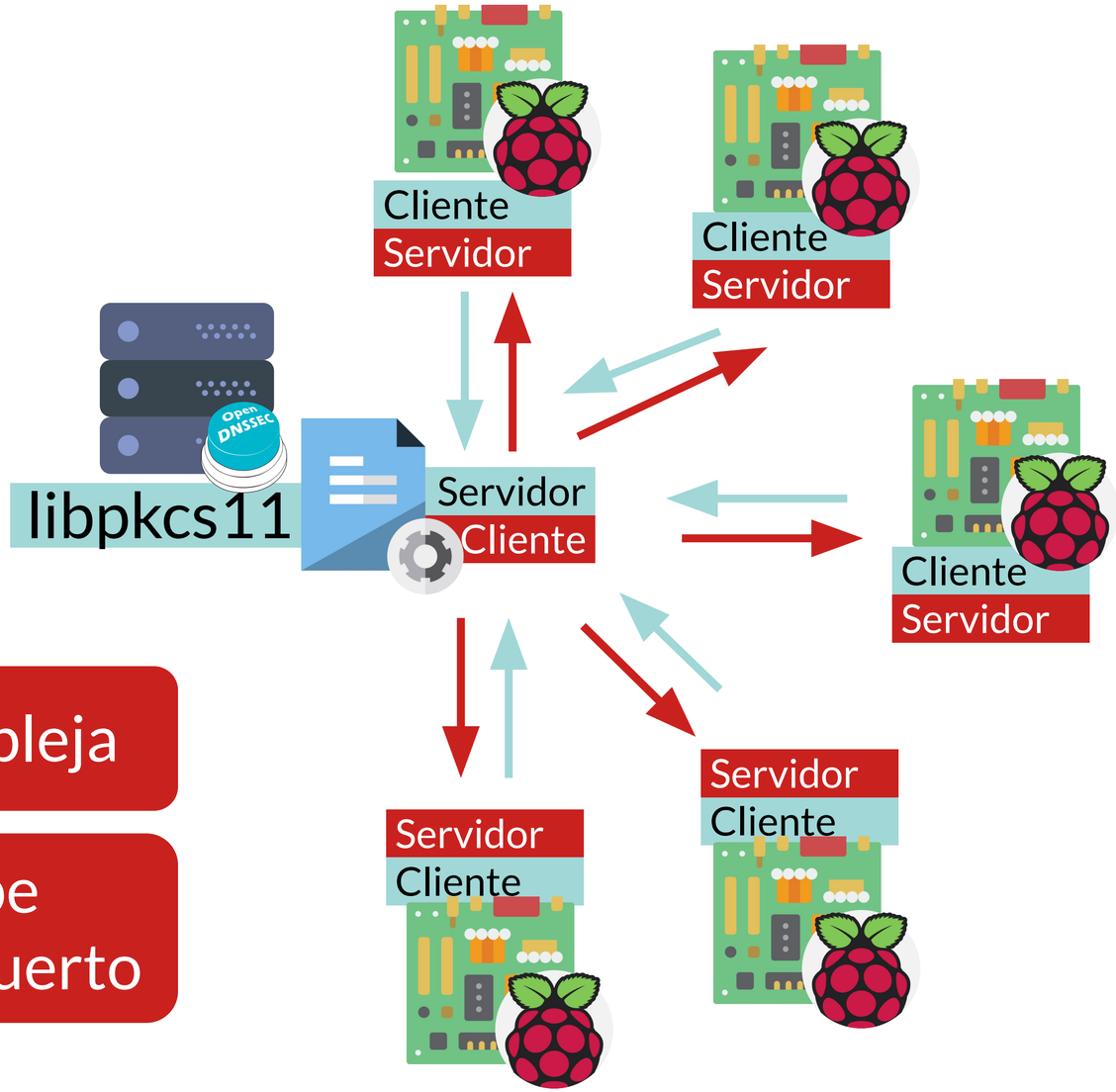
Historia

2016

Implementación en C++ sin nodo maestro (Librería actúa como cliente y servidor)

Instalación compleja

Firmador debe escuchar en un puerto

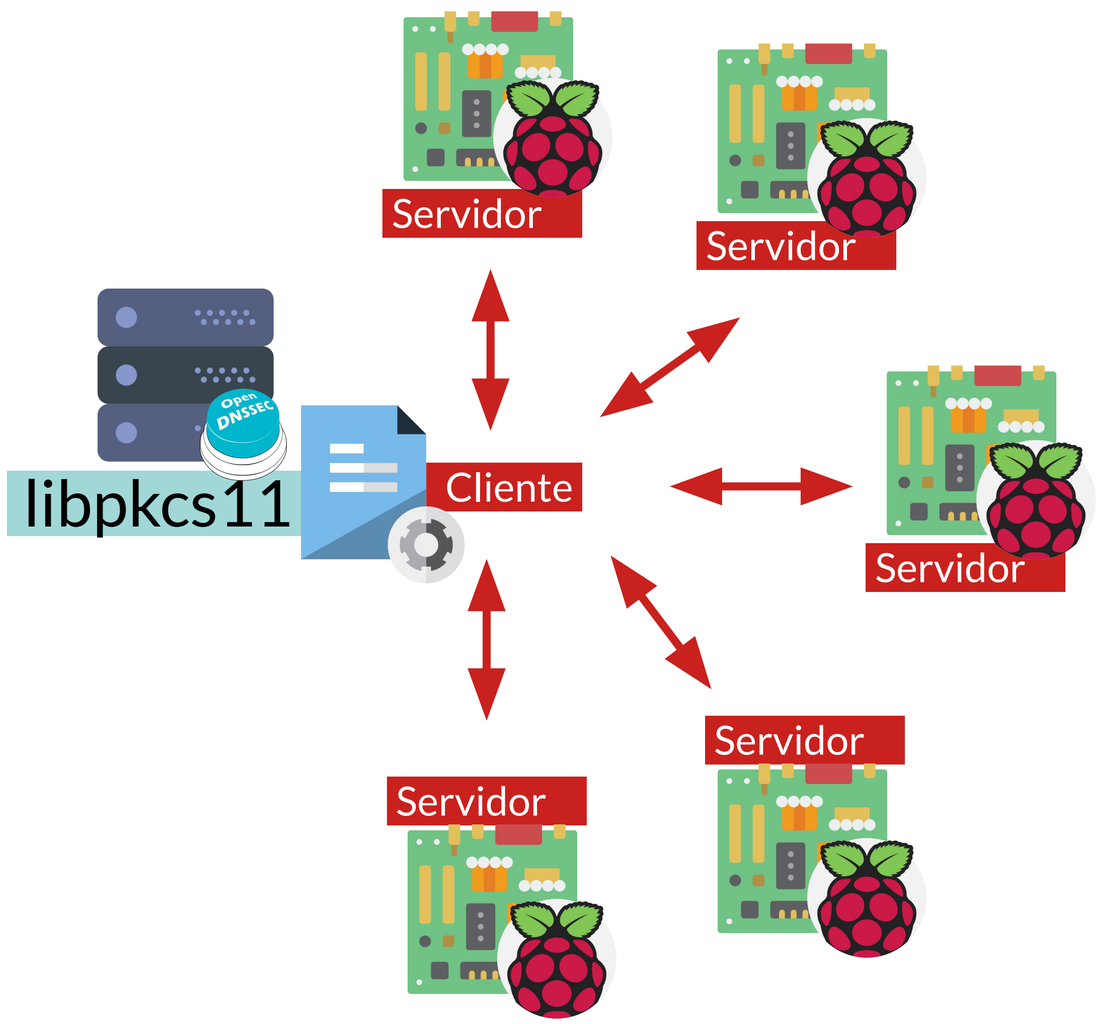


NIC CHILE RESEARCH LABS

Historia

2019

Re implementación en Go.
Instalación vía Go Modules.
Eliminación del servidor en librería.



NIC CHILE RESEARCH LABS

Componentes 2019

Todos disponibles para descarga en
https://niclabs.cl/tchsm

TCRSA

Usa solo librerías nativas de Go (math/big)

DTCNODE

Proceso que corre en nodos

DTC

Compila como librería PKCS11

DHSM-SIGNER

Firmador de zonas en Go

NIC CHILE RESEARCH LABS

Características

Tiempo Compilación

~3 minutos

Tiempo de firmado

~100 firmas/segundo

Generación de Llaves

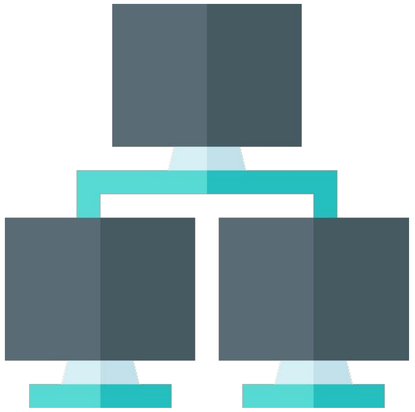
~10s RSA/1024
~3m RSA/2048

NIC CHILE RESEARCH LABS

Trabajo Futuro

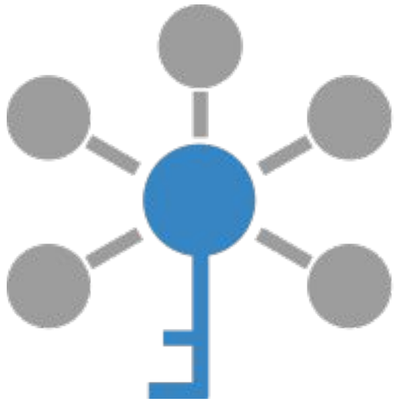
Otros esquemas criptográficos

(que soporten criptografía de umbral)



Otros canales de comunicación

(Aparte de ZMQ)



Probar otros usos de TC/dHSM

No solo para firmado DNSSEC

NIC CHILE RESEARCH LABS

Demostración

NIC CHILE RESEARCH LABS

Agradecimientos

Fco. Cifuentes y Fco. Montoto

Implementación y diseño de primeras versiones del DTCHSM.

Flaticon

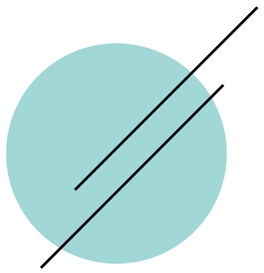
Íconos de Smashicons y Freepik, disponibles bajo licencia Creative Commons en https://flaticon.com

NIC CHILE RESEARCH LABS

Firmado de zonas con el HSM distribuido

Eduardo Riveros
eduardo@niclabs.cl

https://niclabs.cl/tchsm



1. Instalación de Go en nodos y cliente

- https://golang.org/doc/install

Terminal

```
$ wget https://dl.google.com/go/go1.12.9.
     linux-amd64.tar.gz

$ tar -C /usr/local -xzf go1.12.9.
                          Linux-amd64.tar.gz

$ export PATH=$PATH:/usr/local/go/bin
```

NIC CHILE RESEARCH LABS

2. Compilación de librería (DTC)

- Descargarla de Github (Revisar última branch en repositorio)
- Ejecutar ./build.sh, esto generará un archivo llamado dtc.so (es la librería).
- Esperar a que se descarguen las dependencias

Terminal

```
$ git clone https://github.com/niclabs/dtc
--branch v2.0.4

$ cd dtc

$ ./build.sh
```

3. Compilación de librería (DTCNode)

- Descargarla de Github (Revisar última branch en repositorio)
- Ejecutar go build
- Esperar a que se descarguen las dependencias

Terminal

```
$ git clone
https://github.com/niclabs/dtcnode
--branch v2.0.1

$ cd dtcnode

$ go build
```

4. Configuración

- En caso de DTC, basarse en configuración del README del repositorio.
- Crear archivo de configuración en /etc/dtc/config.yaml
- Cambiar nodesNumber y threshold según sea necesario.
- (Configuración continúa en siguiente diapositiva)

Archivo /etc/dtc/config.yaml

```
general:
  logfile: /tmp/dtc.log
  dtc:
    messagingType: zmq
    nodesNumber: 5
    threshold: 3
  criptoki:
    manufacturerId: "NICLabs"
    model: "TCHSM"
    description: "Implementación de PKCS11"
    serialnumber: "1"
    minPinLen: 3
    maxPinLen: 10
    maxSessionCount: 5
    databaseType: sqlite3
    slots:
      - label: TCBHSM
        PIN: 1234
sqlite3:
  path: db.sqlite3 # ubicación DB TCHSM.
...
```

NIC CHILE RESEARCH LABS

5. Configuración

- Dejar pendiente los campos de publicKey/privateKey por ahora.
- Llenar los campos **host** y **port** de cada nodo con la IP y el puerto en el que correrá el servicio de firmas (Por defecto el puerto es 2030)

Archivo /etc/dtc/config.yaml

```
zmq:
  timeout: 10
  publicKey: "..."
  privateKey: "..."
  nodes:
    - host: 5.6.7.8
      publicKey: '...'
      port: 2030
    … # Tantas entradas como nodos en el
sistema
```

NIC CHILE RESEARCH LABS

6. Configuración

- Con ayuda de DTCNode, crear un nuevo par de llaves y colocarlo en la configuración de la librería (/etc/dtc/config.yaml)

  NO REUTILIZAR LAS LLAVES DE EJEMPLO!

Terminal

```
$ ./dtcnode generate-curve
> Public Key:   (T&I$J^5(vYpx%J4?jq...
> Private Key:  HvN8<lQ:VI[AwS2Jgat...
```

NIC CHILE RESEARCH LABS

7. Configuración e inicio de nodos

- Crear los archivos de configuración de cada nodo con el comando de **dtcnode**.
- Usar en “-c” la IP que verán los nodos cuando el cliente use la librería (en ejemplo es 1.2.3.4)
- Usar en “-k” la llave pública que usará la librería al conectarse.
- Correr nodos.

Terminal

```
$ ./dtcnode generate-config -c 1.2.3.4 -k
(T&I$J^5(vYpx%J4?jq -o config.yaml

$ ./dtcnode serve
```

NIC CHILE RESEARCH LABS

8. Configuración de los nodos

- Contiene los campos fijados en el paso anterior.
- Copiar llaves “publicKey” de cada nodo en el archivo de configuración de la librería.

config.yaml (en carpeta dtcnode)

```
config:
  publickey: '@?Bpu79j8JG1...'
  privatekey: ‘jlACUxB<j((...’
  host: 0.0.0.0
  port: 2030
  client:
     publickey: '(T&I$J^5(vYpx%J4?jq...'
     host: localhost
    Keys: []
```

9. Para usar con dhsm-signer

- Setear como flag **p** la ruta a la librería compilada.
- En caso de dudas con las flags, ejecutar **dhsm-signer** y ver la ayuda del comando.

Terminal

```
$ git clone https://github.com/niclabs/
          dhsm-signer -v1.0
$ cd dhsm-signer
$ go build
$ ./dhsm-signer sign -p ruta/dtc.so -f
example.com -o example.com.signed -z
example.com -3 -c
```

Para usar con OpenDNSSEC

- Editar /etc/opendnssec/conf.xml
- Modificar o agregar un nuevo repositorio, con un valor “Module” que apunte a la librería compilada en el paso 2.
- Colocar en **TokenLabel** y **Pin** los mismos valores que en la configuración de la librería.

/etc/opendnssec/conf.xml

```
...
  <RepositoryList>

     <Repository name="SoftHSM">
       <Module>ruta/dtc.so</Module>
       <TokenLabel>TCBHSM</TokenLabel>
       <PIN>1234</PIN>
       <SkipPublicKey/>
       <!--
       <AllowExtraction/>
       -->
     </Repository>
...
```

NIC CHILE RESEARCH LABS